J. Mod. Transport. (2015) 23(1):67–79
DOI 10.1007/s40534-014-0063-x

Relative efficiency appraisal of discrete choice modeling algorithms using small-scale maximum likelihood estimator through empirically tailored computing environment

Hyuk-Jae Roh · Prasanta K. Sahu · Ata M. Khan · Satish Sharma

Received: 22 August 2014 / Revised: 31 October 2014 / Accepted: 3 November 2014 / Published online: 20 November 2014

Abstract Discrete choice models are widely used in multiple sectors such as transportation, health, energy, and marketing, etc., where the model estimation is usually carried out by using commercial software. Nonetheless, tailored computer codes offer modellers greater flexibility and control of unique modelling situation. Aligned with empirically tailored computing environment, this research discusses the relative performance of six different algorithms of a discrete choice model using three key performance measures: convergence time, number of iterations, and iteration time. The computer codes are developed by using Visual Basic Application (VBA). Maximum likelihood function (MLF) is formulated and the mathematical relationships of gradient and Hessian matrix are analytically derived to carry out the estimation process. The estimated parameter values clearly suggest that convergence criterion and initial guessing of parameters are the two critical factors in determining the overall estimation performance of a custom-built discrete choice model.

H.-J. Roh (✉)
City of Regina, Regina, SK S4P 3C8, Canada
e-mail: hyukjae.roh@gmail.com

P. K. Sahu
National Institute of Construction Management and Research, Pune 411045, India
e-mail: prasantsahu222@gmail.com

A. M. Khan
Department of Civil and Environmental Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada
e-mail: ata.khan@carleton.ca

S. Sharma
Environmental Systems Engineering, University of Regina, Regina, SK S4S 0A2, Canada
e-mail: Satish.Sharma@uregina.ca

Keywords Estimation algorithms · Visual basic application · Convergence criterion · Binary logit · Maximum likelihood

1 Introduction

It is common practice to carry out mathematical modeling for transportation planning, design, and operation by using readily available commercial software packages. Generally, the estimation processes are embedded in the software while developing the software package. This leads to constraints for users due to lack of information on the underlying analytical procedure for the model parameter estimation. This practice also applies to discrete choice model parameter estimation. There has been a general consensus that transportation modellers do not need to put extra efforts to come up with tailored computer codes dedicated solely for their unique modelling need. This may work in most cases; however, it may fail to work for very specific cases. In this context, the lack of flexibility in the computing environment could be exemplified through a situation, where the researchers choose 'non-linear-in-parameters' specification. Commercial packages cannot handle such a specification, and thus, the researcher might be forced to write a specific computer code dedicated only for that work [1, 2].

Advances in statistical computing language are readily available and could be applied as modeling base to all estimation procedures in a systematic way. A number of suitable computing languages for model estimation have improved significantly over the past years from, when reliance was almost exclusively on commercial software. As the statistical computing language of modeling has evolved significantly, more sophisticated modeling procedures could now be tailored by researchers in accordance

with the level of transport modeling analysis. In an effort to relax limitations of software, researchers are strongly encouraged to use their own modeling computer codes [3]. On the basis of a clear understanding of analytical procedures that are repeatedly taking place during the estimation processes, researchers would be able to use different estimation algorithms and empirically modify/improve some portions of estimation processes. Eventually, with diverse numerical experiments, this repetitive task will help in proposing a new advanced/improved estimation procedure based on certain performance criterion.

Additionally, transportation researchers have not paid much attention on how to go beyond the limitation of using commercial packages in terms of carrying out entire modelling procedures with tailored computing environments [4]. This purpose is pursued in this research with a small scale discrete choice model, namely the binary logit model. The main objective of this research is to implement all analytical procedures involved in discrete choice modelling with visual basic application (VBA) and to present results of numerical experiments solely through possible tailored modeling. The other objectives of this research are (a) to identify the factors affecting the model estimation performance and (b) to suggest potential research topics related to custom-built modeling.

2 Derivation of multinomial logit

This section presents a binary logit choice function derived from the multinomial logit (MNL) choice function. The binary logit model is used as the target choice function for experimentation in this research. The derivation procedure adopted here helps in understanding not only the model's limitations but also the conceptual background based on which the model is built. Consequently, researchers can differentiate MNL from other types of discrete choice models with utmost clarity and may try with a variety of experimental model estimation trials. Based on the research by Train [3] and Ben-Akiva and Lermans [5], the algebraic manipulation to get a final logit choice function is described here in details. MNL model, which is a prototype model of discrete choice models, is formulated based on the random utility concept. The probability that any alternative in C_n is chosen by decision makers can be formulated as in Eq. 1. C_n is actual choice set that the decision maker (n) can consider in the given choice situation.

$$P_n(i) = P(U_{ni} > U_{nj}, \quad \forall j \in C_n, i \neq j), \tag{1}$$

where $P_n(i)$ is the probability of choosing a alternative i by decision maker n; U_{ni} and U_{nj} are representative utility of the alternatives i and j of the decision n, $\forall j \in C_n$, $i \neq j$ indicates comparison of all alternatives except for the same alternative. Based on assumption, the utility can be divided into systematic and random parts (shown in Eq. 2).

$$P_n(i) = P\left(S_{ni} + R_{ni} > S_{nj} + R_{nj}, \forall j \in C_n, i \neq j\right), \tag{2}$$

$$= P\left(R_{nj} < S_{ni} - S_{nj} + R_{ni}, \forall j \in C_n, i \neq j\right), \tag{3}$$

where S_{ni} is the systematic (or deterministic, observable, constant) part of the representative utility of alternative i when faced by decision maker n and is observable to a researcher, and R_{ni} is the random (unobservable, error) part of the representative utility of alternative i when faced by decision maker n and is unobservable to a researcher. Usually, various discrete choice models are derived from Eq. (3), which depends on the mathematical distribution (e.g., Gumbel Type I, and Weibull, etc.)that the random component (R_{ni}, R_{nj}) follows. Therefore, using density function of Gumbel Type I (or Weibull), the density for each random parts relation is shown in Eq. 4 and the cumulative distribution is presented in Eq. 5.

$$f(R_{na}) = e^{-R_{na}} e^{-e^{-R_{na}}}, \tag{4}$$

$$F(R_{na}) = e^{-e^{-R_{na}}}. \tag{5}$$

In Eq. (3), if R_{ni} is not known, then the choice probability of $P_n(i)$ is a cumulative distribution, whose value is determined at $S_{ni}- S_{nj} + R_{ni}$, as given in Eq. (5).

$$F\left(R_{nj}\right) = e^{-e^{-(S_{ni}-S_{nj}+R_{ni})}}, \tag{6}$$

where R_{nj} is distributed independently for all alternatives, $i \neq j$ and it is logical that the Eq. 3 means that the choice probability is the product of the individual value of cumulative distribution obtained using Eq. 6. Equation 7 presents this function.

$$P_n(i)|R_{ni} = \prod\nolimits_{j \neq i} e^{-e^{-(S_{ni}-S_{nj}+R_{ni})}}. \tag{7}$$

As mentioned earlier, R_{ni} is unknown and is assumed to have a distribution. By integration of $P_n(i)|R_{ni}$, $P_n(i)$ will be determined as shown in Eq. 8.

$$P_n(i) = \int \left(\prod\nolimits_{j \neq i} e^{-e^{-\left(S_{ni}-S_{nj}+R_{ni}\right)}}\right)\left(e^{-R_{ni}} e^{-e^{-R_{ni}}}\right) \mathrm{d}R_{ni}. \tag{8}$$

Some more algebraic interpolations need to be made to reach the final form of logit choice function. The remained processes are given below:

$$P_n(i) = \int_{R_{ni}=-\infty}^{\infty} \left(\prod\nolimits_{j \neq i} e^{-e^{-\left(S_{ni}-S_{nj}+R_{ni}\right)}}\right)\left(e^{-R_{ni}} e^{-e^{-R_{ni}}}\right) \mathrm{d}R_{ni}, \tag{9}$$

$$= \int_{R_{ni}=-\infty}^{\infty} \left(\prod\nolimits_{j \neq i} e^{-e^{-\left(S_{ni}-S_{nj}+R_{ni}\right)}}\right)\left(e^{-R_{ni}} e^{-e^{-(S_{ni}-S_{ni}+R_{ni})}}\right) \mathrm{d}R_{ni}, \tag{10}$$

$$= \int_{R_{ni}=-\infty}^{\infty} \left(\prod\nolimits_{j} e^{-e^{-\left(S_{ni}-S_{nj}+R_{ni}\right)}}\right)\left(e^{-R_{ni}}\right) \mathrm{d}R_{ni}, \tag{11}$$

$$= \int_{R_{ni=-\infty}}^{\infty} \left(e^{-e^{-(S_{ni}-S_{n1}+R_{ni})}} e^{-e^{-(S_{ni}-S_{n2}+R_{ni})}} \cdots e^{-e^{-(S_{ni}-S_{ni}+R_{ni})}} \right) \left(e^{-R_{ni}} \right) \mathrm{d}R_{ni}, \tag{12}$$

$$= \int_{R_{ni=-\infty}}^{\infty} \left(e^{-e^{-(S_{ni}-S_{n1}+R_{ni})} - e^{-(S_{ni}-S_{n2}+R_{ni})} \ldots - e^{-(S_{ni}-S_{ni}+R_{ni})}} \right) \left(e^{-R_{ni}} \right) \mathrm{d}R_{ni}. \tag{13}$$

The Eq. (13) can be simplified as follows:

$$P_n(i) = \int_{R_{ni=-\infty}}^{\infty} \left(e^{-\sum_j e^{-(S_{ni}-S_{nj}+R_{ni})}} \right) \left(e^{-R_{ni}} \right) \mathrm{d}R_{ni}, \tag{14}$$

$$= \int_{R_{ni=-\infty}}^{\infty} \left(e^{-e^{-R_{ni}} \sum_j e^{-(S_{ni}-S_{nj})}} \right) \left(e^{-R_{ni}} \right) \mathrm{d}R_{ni}. \tag{15}$$

Let $e^{-R_{ni}} = Z$, and after differentiating with respect to R_{ni}, it becomes $-e^{-R_{ni}} \mathrm{d}R_{ni} = \mathrm{d}Z$. It is noted that if $R_{ni} \to \infty$ then, $Z \to \infty$, if $R_{ni} \to \infty$ then $Z \to 0$. Replacing these terms in Eq. 15, the final logit choice function is obtained as below (See Eq. 18).

$$P_n(i) = \int_{Z=0}^{\infty} e^{-Z \sum_j e^{-(S_{ni}-S_{nj})}} \mathrm{d}Z, \tag{16}$$

$$= \left[\frac{e^{-Z \sum_j e^{-(S_{ni}-S_{nj})}}}{-\sum_j e^{-(S_{ni}-S_{nj})}} \right]_0^{\infty}, \tag{17}$$

$$= \frac{e^{S_{ni}}}{\sum_j e^{S_{nj}}}. \tag{18}$$

The final form of probabilistic MNL model consists of two parts. The denominator represents the total utility of all the alternatives considered in the choice set and theutility of a particular alternative is given in the numerator. The binary logit model is utilized as numerical experimental base in this research, which can be simplified from Eq. 18 by assuming that only two alternatives are available in the mode choice market. Assuming two alternatives (auto and transit) are available; the Eq. 18 can be simplified into the binary logit choice formulation as shown in Eq. 19, where S_{na} and S_{nt} are the systematic part of the representative utility of alternative auto and transit, respectively.

$$P_n(autoortransit) = \frac{e^{S_{na}} (or\, e^{S_{nt}})}{e^{S_{na}} + e^{S_{nt}}}. \tag{19}$$

3 Methodology

3.1 Data, utility, likelihood, and log-likelihood function

Numerical experiments, in this research, are carried out using simple database that are successfully applied by Ben-Akiva and Lermans [5] for model parameter estimation of a given model specification. The reason behind this is, that a proven data base provides guarantee for successful convergence of the experimental model run. Additionally, the value of parameter estimates generated through experimental runs can be compared to true values and hence it may be concluded that all experimental runs are conducted on the right estimation track. Data plugged in model specification within tailored computing environments allow research to keep track of computational efforts: such as log-likelihood values generated for each observation until model convergence is achieved.

The model specification exemplified in a Ref. [5] is adopted in this research in order to compare models in the exact same conditions. Two variables are included in model specification. The first one is a generic variable belonging to both modes (i.e., auto and transit) and the other one is a mode-specific constant for auto mode (since the transit mode is designated as the reference mode). In other words, T_A and T_T are included as service-level attributes in the utility function for these two modes. Consequently, the final model specifications are shown as below in Eqs. 20 and 21

$$V_{An} = \beta_1 + \beta_2 T_A, \tag{20}$$

$$V_{Tn} = \beta_2 T_T. \tag{21}$$

where V_{An} and V_{Tn} are the utility functions of auto and Transit. β_1 and β_2 are model parameters to be estimated in the computation process.

In the next step, we considered the estimation method, which is one of core parts in custom-built modeling to get an in-depth understanding about the data involvement in discrete choice analysis. We applied the maximum likelihood function throughout the research: developing the computer code, deriving formulas for elements of Hessian matrix, and Gradient vector etc. Newton–Raphson estimation algorithm is employed mainly because of its excellence in convergence characteristics, and, more importantly, its higher level of accuracy for the parameter estimated. Data used for this research is assumed to be selected at random from a population and the likelihood function of the entire sample is the product of the likelihoods of each observation. Therefore the final form of the function specified for this research is given in Eq. 22. We used the log-likelihood function for the ease of computation and this is given in Eq. 23. The binary logit choice function tailored formulations for the two modes (auto, transit) are given in Eq. 24.

$$L(\beta_1, \beta_2) = \prod_{n=1}^{21} P_n(A)^{y_{An}} P_n(T)^{y_{Tn}}, \tag{22}$$

$$LL(\beta_1, \beta_2) = \sum_{n=1}^{21} [y_{An} ln P_n(A) + y_{Tn} ln P_n(T)], \tag{23}$$

$$P_n(A) = \frac{e^{V_{An}}}{e^{V_{An}} + e^{V_{An}}}, \; P_n(T) = \frac{e^{V_{Tn}}}{e^{V_{An}} + e^{V_{An}}}. \tag{24}$$

Replacing Eqs. 20, 21 and 24 in 23, the final form of log-likelihood function specified for this experimental study is presented in Eq. 25. This equation is used as a base equation for obtaining a variety of formulas with which a tailored-computer code is developed.

$$LL(\beta_1,\beta_2)=\sum_{n=1}^{21}\left[y_{An}\ln\left(\frac{e^{\beta_1+\beta_2 T_A}}{e^{\beta_1+\beta_2 T_A}+e^{\beta_2 T_T}}\right)+y_{Tn}\ln\left(\frac{e^{\beta_2 T_A}}{e^{\beta_1+\beta_2 T_A}+e^{\beta_2 T_T}}\right)\right]. \tag{25}$$

.

3.2 Derivation of gradient and Hessian formulas

The gradient of the log-likelihood function, which is a vector of the first derivatives of Eq. 25, plays an important role in determining the direction for parameter estimation. We derived the first-order partial derivatives of the Eq. 25; knowing the importance of the estimation procedure for developing the computer code. The most critical part of developing a computer program for estimation is to understand the relationship between the data and derived formulations used for estimation. Particularly, in discrete choice model, the functional form of the model looks simple; however mapping a relationship between the formulae and data set from the operational perspective is vital to define. In addition, the expansion of the given log-likelihood function is computationally extensive and costly [4]. In this section, we have presented only the final formulation of the gradient vector, which has been derived by using partial derivatives in analytical terms. The elements included in a gradient vector can be described as below in the mathematical form by Eqs. 26 and 27.

$$\frac{\partial LL(\beta_1,\beta_2)}{\partial\beta_1}=\sum_{n=1}^{21}[y_{An}P_n(T)-y_{Tn}P_n(A)], \tag{26}$$

$$\frac{\partial LL(\beta_1,\beta_2)}{\partial\beta_2}=\sum_{n=1}^{21}[y_{An}P_n(T)(T_{An}-T_{Tn})-y_{Tn}P_n(A)(T_{Tn}-T_{An})]. \tag{27}$$

We calculated the real gradient values through the estimation process and the value is constantly updated by new values until iteration process is terminated with final parameters estimates. A gradient vector of log-likelihood function can be summarized in a vector (column vector) form as shown on Eq. 28. In the context of this research, the column vector includes the two explanatory variables.

$$G_t=\frac{\partial LL(\beta_1,\beta_2)}{\partial\beta_k}=\begin{bmatrix}\frac{\partial LL(\beta_1,\beta_2)}{\partial\beta_1}\\ \frac{\partial LL(\beta_1,\beta_2)}{\partial\beta_2}\end{bmatrix}=\begin{bmatrix}\sum_{n=1}^{21}[y_{An}P_n(T)-y_{Tn}P_n(A)]\\ \sum_{n=1}^{21}[y_{An}P_n(T)(T_{An}-T_{Tn})-y_{Tn}P_n(A)(T_{Tn}-T_{An})]\end{bmatrix}. \tag{28}$$

Finally, the mathematical expression of a gradient vector can be translated into computer code based on the interpretation implied in the above formula expansion and its numerical value is continuously changed during iterations until the process is terminated.

In order to employ Newton–Raphson estimation algorithm as an iteration method in custom-built discrete choice modeling, it is needed to derive the second-order partial derivatives of the log-likelihood formulation for the given choice situation. Therefore it becomes obvious for the researchers (or modellers) to derive analytically the second-order partial derivatives of the log-likelihood function [4], which contains full information associated with sample used in the analysis. Indeed, the process involved in the analysis is extremely troublesome and computationally intensive; however, there is no commercial software that offers the same level of understanding to researchers. Also, the experience gained practically in a way that all procedures associated with model estimation are mimicked analytically. In this context, the Newton–Raphson algorithm combined with the real Hessian matrix shows the best estimation performance in discrete mode choice modeling compared to any other algorithm operated by approximate Hessian matrix such as BHHH, BHHH-2, BFGS to name a few [3]. Based on the knowledge of the Hessian matrix symmetry, only the off-diagonal elements of Hessian matrix are treated mathematically. The following part of this section is mainly focused on the description of the formulas resulted from the second-order partial derivatives of the log-likelihood function for the off-diagonal elements of the Hessian matrix. For the purpose of simplicity, off-diagonal elements included in the Hessian matrix is formulated in the forms of relations given by Eqs. 29,30 and 31 by calculating the second-order derivatives with respect to $\beta_1\beta_1$, $\beta_2\beta_2$ (or $\beta_1\beta_1$ and $\beta_2\beta_2$). The formulations are shown below.

$$\frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_1\partial\beta_1}=\sum_{n=1}^{21}[-y_{An}P_n(T)P_n(A)-y_{Tn}P_n(A)P_n(T)], \tag{29}$$

$$\frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_1\partial\beta_2(or\partial\beta_1\partial\beta_2)}=\sum_{n=1}^{21}[y_{An}P_n(T)P_n(A)(T_{Tn}-T_{An})-y_{Tn}P_n(T)P_n(A)(T_{An}-T_{Tn})], \tag{30}$$

$$\frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_2\partial\beta_2} = \sum_{n=1}^{21}\left[-y_{An}P_n(T)P_n(A)(T_{An}-T_{Tn})^2 - y_{Tn}P_n(T)P_n(A)(T_{An}-T_{Tn})^2\right] \tag{31}$$

.

The three elements consist of Hessian matrix is given in Eq. 32.

$$H_t = \begin{bmatrix} \frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_1\partial\beta_1} & \frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_1\partial\beta_2} \\ \frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_2\partial\beta_1} & \frac{\partial^2 LL(\beta_1,\beta_2)}{\partial\beta_2\partial\beta_2} \end{bmatrix}. \tag{32}$$

Finally, all prerequisites that researchers need before the custom-built discrete choice modeling process are prepared.

3.3 Visual basic application (VBA)

Computer codes are written for estimation algorithms and the maximum likelihood method that are explained in the previous section. The VBA package in Microsoft Excel was utilized for this purpose. It is notable to know that there are two critical parts for researchers to be concerned about in developing computer codes. The first is to know how the log-likelihood function computer codes could be realized and the second is to know how the estimation algorithms could be included in the language of computer codes. All estimation runs are conducted using a personal computer (Intel Pentium 4, CPU 2.80 GHz, 512 MB Ram).

4 Iteration algorithms and estimation factors

In off-the-shelf software application for modeling, the software primarily enables a researcher to find the optimal set of parameter estimates under a given estimation algorithm, where, estimation factors are prefixed by the vendor. However in custom-built modeling environments, finding parameter estimates typically depends on iteration method predefined earlier in chosen estimation algorithm. Furthermore, to make the estimation operational, a researcher will be directed to input initial estimation factors which would affect the overall estimation performance at the initial stage of estimation. These factors include step size and convergence criterion. Six algorithms are applied to experimentally estimate parameters with a given model specification in the binary logit choice model framework. It is noticed during the course of this research, that the iteration method involved in each six algorithms mutates each other than updating process of approximate Hessian. Therefore, we discussed in detail the role of estimation factors including the estimation method based on the Newton–Raphson (NR) algorithm in the following paragraphs.

Newton–Raphson algorithm uses the real Hessian matrix in its estimation routine by employing true Hessian values generated from the second-derivative of log-likelihood function. The computations for obtaining real Hessian matrix are expensive and costly. Moreover, majority of work associated with the analytical derivation of log-likelihood function are extensive, nevertheless it is very useful [4]. In this research, the mathematical relationships for the real Hessian matrix are formulated by means of direct analytical derivatives from the given log-likelihood function (Eq. 25) and the final form of matrix is provided through the Eqs. 29–31. Translating estimation procedures into computer codes involves: 1) understanding the fundamental iteration rule for Newton–Raphson algorithm, and 2) separating the functionality of estimation procedures into independent, interchangeable modules. The estimation method of Newton–Raphson employed in this research can be derived by taking the second order Taylor's approximation of $LL(\beta_{t+1})$ around $LL(\beta_t)$ and the details are as follows. The Taylor's series expansion of $f(x)$ around $f(x_0)$ can be written as below [4].

$$f(x) = \frac{f(x_0)}{(0)!} + \frac{f^{(1)}(x_0)}{(1)!}(x-x_0) + \frac{f^{(2)}(x_0)}{(2)!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{(n)!}(x-x_0)^n. \tag{33}$$

The above function can be shortened by taking the second-order approximation of Taylor's series as shown below:

$$LL(\beta_{t+1}) \cong \frac{LL(\beta_t)}{(0)!} + \frac{LL^{(1)}(\beta_t)}{(1)!}(\beta_{t+1}-\beta_t) + \frac{LL^{(2)}(\beta_t)}{(2)!}(\beta_{t+1}-\beta_t)^2, \tag{34}$$

$$= LL(\beta_t) + G_t(\beta_{t+1}-\beta_t) + \frac{1}{2}H_t(\beta_{t+1}-\beta_t)^2. \tag{35}$$

It should be noted that $f(x)$ is $LL(\beta_{t+1})$ and $f(x_0)$ is $LL(\beta_t)$ in Eq. (34). Based on the Eq. (35), the maximization value of β_{t+1} can be found by setting $\frac{\partial LL(\beta_{t+1})}{\partial\beta_{t+1}} = 0$. The first derivatives with respect to β_{t+1} results in

$$\frac{\partial LL(\beta_{t+1})}{\partial\beta_{t+1}} = G_t + H_t(\beta_{t+1}-\beta_t). \tag{36}$$

Consequently, by solving the Eq. 36 for the value of β_{t+1}, the iteration rule of Newton–Raphson is represented by the following formula:

$$\beta_{t+1} = \beta_t + \lambda(-H_{tReal}^{NR})^{-1} G_t, \quad (37)$$

where β_t is the parameter estimates after t iterations, β_{t+1} is the parameter estimates after $t + 1$ iterations, λ is the step size, $(-H_{tReal}^{NR})^{-1}$ is the inverse of the negative real Hessian matrix in iteration t, G_t is a vector of gradient in iteration t, $(-H_{tReal}^{NR})^{-1} G_t$ is the search direction of the function in iteration process. Stated more simply, β_{t+1} which is $(t + 1)$ th parameter estimates is equal to β_t which is tth parameter estimates plus negative inverse Hessian multiplied by gradient. A way of the estimation method of other five algorithms considered in this research can be found in detail in the previous study conducted by Roh and Khan [4], Train [3], and Green [6].

In this study, both H_{tReal}^{NR} and G_t are the average value of the Hessian and gradient. The Hessian and gradient are calculated at each step and continuously updated till the convergence. Based on the fact that the log-likelihood function of logit probability is always globally concave [7], the following paragraph explains how parameters are searched during iteration process from the initial guessing points to final convergence of parameter estimates.

The "the direction to follow" concept in the context of iteration method is shown in Fig. 1. G_t is the slope of the log-likelihood function and H_{tReal}^{NR} shows the change in slope of the log-likelihood function. In cases of global concave function, H_{tReal}^{NR} is always negative and thus $(H_{tReal}^{NR})^{-1}$ is also negative. The negative of this negative Hessian, $-(H_{tReal}^{NR})^{-1} G_t$ is positive. Therefore, the direction of $-(H_{tReal}^{NR})^{-1} G_t$ depends entirely on the sign of G_t. If the G_t is positive, β_t moves to the right direction (Fig. 1a), otherwise, β_t moves to the left direction (Fig. 1b) according to the iteration rule seen in Eq. 37. Its step size is given by a researcher at the initial stage of iteration and its magnitude is given by the term, $\lambda(-H_{tReal}^{NR})^{-1} G_t$, included in the iteration rule. In either case, β_t is moved to the best value that maximizes the log-likelihood function.

The step size implied by $\lambda(-H_{tReal}^{NR})^{-1} G_t$ is decreased, if the curvature represented by $-(H_{tReal}^{NR})^{-1}$ has a large value in its magnitude (Fig. 2a); otherwise, the step size is increased (Fig. 2b).

For understanding "step size" in the context of iteration method, it is useful to know the maximization of $LL(\beta)$:$(LL(\beta) = A + B\beta + C\beta^2)$, and it's convergence to

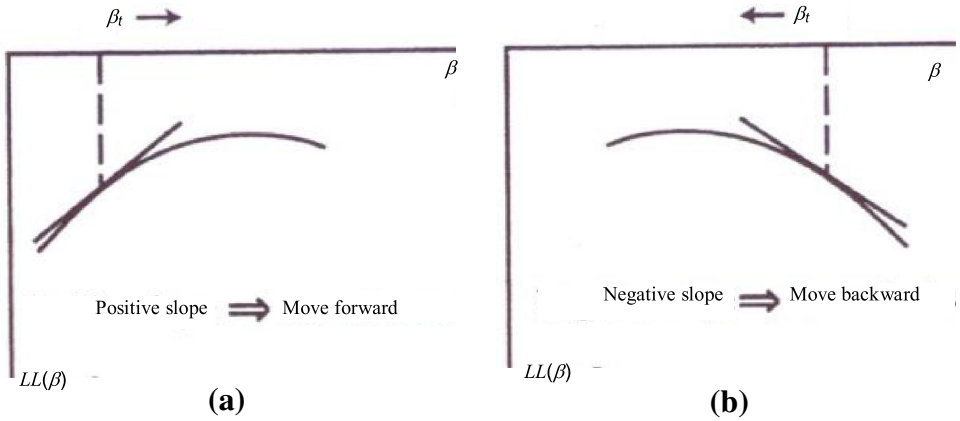

Fig. 1 Direction of step follows the slope [3]

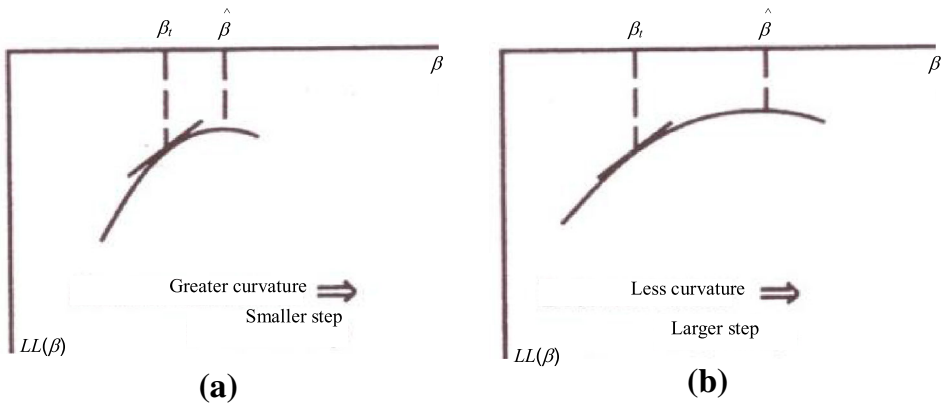

Fig. 2 Step size is inversely related to curvature [3]

maximum. In the case of quadratic function, the optimum value can be obtained with one iteration [3]. However, the problem is that most $LL(\beta)$ are not quadratic in nature and they require more than one iteration to attain the optimum. This problem can be solved by choosing an appropriate initial step size, and hence minimizing the iteration number and convergence time. The step size determination is explained in the subsequent paragraph.

4.1 Case–I: *LL* (β) is not a quadratic function

Figure 3 shows two different functions represented by solid and dashed lines. The solid line represents the actual $LL(\beta)$ and the quadratic form is represented by the dashed line. After one iteration ($\lambda = 1$), if the given function is quadratic, then β_{t+1} reaches to the maximum point. But, the actual $LL(\beta)$ is not quadratic, thus β_{t+1} goes beyond the maximum point and as a result of which, $LL(\beta_{t+1}) < LL(\beta_t)$. To ensure that $LL(\beta_{t+1}) > LL(\beta_t)$, and to guarantee an increasing $LL(\beta)$, one can decrease λ to 1/2. After this operation, it is logical to compare the values between $LL(\beta_{t+1})$ and $LL(\beta_t)$. If $LL(\beta_{t+1}) < LL(\beta_t)$, then λ may be further decreased to 1/4 The λ value will be decreased to smaller values such as 1/8 and 1/16 repeatedly until $LL(\beta_{t+1}) > LL(\beta_t)$, is satisfied at certain λ.

4.2 Case–II: $LL(\beta)$ is a quadratic function

The second case is completely opposite to the first case. As shown in Fig. 4, the solid line is for actual $LL(\beta)$ and dashed line is for quadratic. At first iteration with $\lambda = 1$, β_{t+1}, reaches to the maximum for the quadratic case. But, the actual $LL(\beta)$ is not quadratic and the maximum point is far away and λ should be adjusted by taking a large step in order to satisfy $LL(\beta_{t+1}) > LL(\beta_t)$. In Fig. 4, it can be seen that $\lambda = 2$ is the best step size to be chosen for estimation process.

Subsequently, VBA tailored computing codes developed earlier are utilized for the six different estimation algorithms to compare their convergence behaviour in terms of

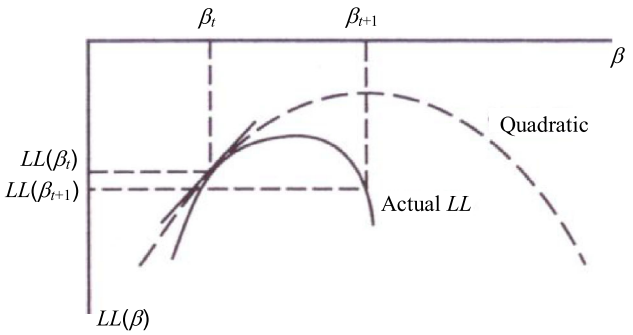

Fig. 3 Finding a proper step size in the first case [3]

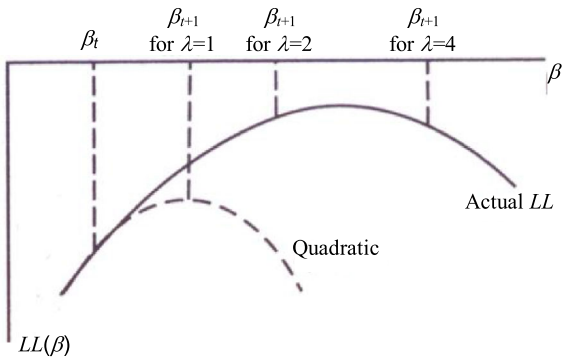

Fig. 4 Finding a proper step size in the second case [3]

performance measures. The following section presents and discusses the results from the experiments.

5 Experimental results

The computer codes developed for this research worked successfully within the simulated discrete choice model estimation environments. The values of parameter estimates of the model are compared with true value [5] and it is found that the values of parameter estimates are closer in magnitude across six algorithms. This means that the computer codes developed for this empirical estimation worked on right estimation track. The parameter estimates for six algorithms are presented in Table 1 with different step sizes, which λ shows the best performance in a given convergence criterion: $CR = 10^{-4}$. It can also be noted that all estimation procedures involved in discrete choice modelling are completely translated into tailored computer estimation codes. The purpose of this research is to get in-depth understanding of calculation mechanism underlying in discrete choice modeling, not to develop a model. Therefore, statistical explanations of parameters are not within the scope of this research.

The relative performances of algorithms are compared by using the three performance measures: number of iteration, iteration time, and convergence time. In the first case, we chose different convergence criteria and the second case is involved with different initial guesses of the parameter values. After applying various step sizes along with the iteration rule, the experimental runs that converged successfully are selected and summarized.

Before presenting experiment results, overall convergence characteristics of six algorithms is presented using 3D convergence surface as shown in Fig. 5. Only a run showing the best performance for each algorithm is depicted in 3D convergence surface. Variation of log-likelihood values for all observations in each iteration are continuously saved in server until the estimation process is terminated. Unique

Table 1 Parameters estimated from six algorithms ($CR = 10^{-4}$)

Variables	NR ($\lambda = 1$)	BHHH ($\lambda = 1/2$)	BHHH-2 ($\lambda = 1/2$)	SA ($\lambda = 16$)	DFP ($\lambda = 16$)	BFGS ($\lambda = 8$)
β_1	−0.237575	−0.237462	−0.237428	−0.237588	−0.237575	−0.237576
β_2	−3.186590	−3.186410	−3.186355	−3.186671	−3.186590	−3.186590

NR Newton–Raphson, *BHHH* Berndt, Hall, Hall and Hausman, *SA* Steepest Ascent, *DFP* Davidon, Fletcher, and Powell, *BFGS* Broydon, Fletcher, Goldfarb, and Shanno

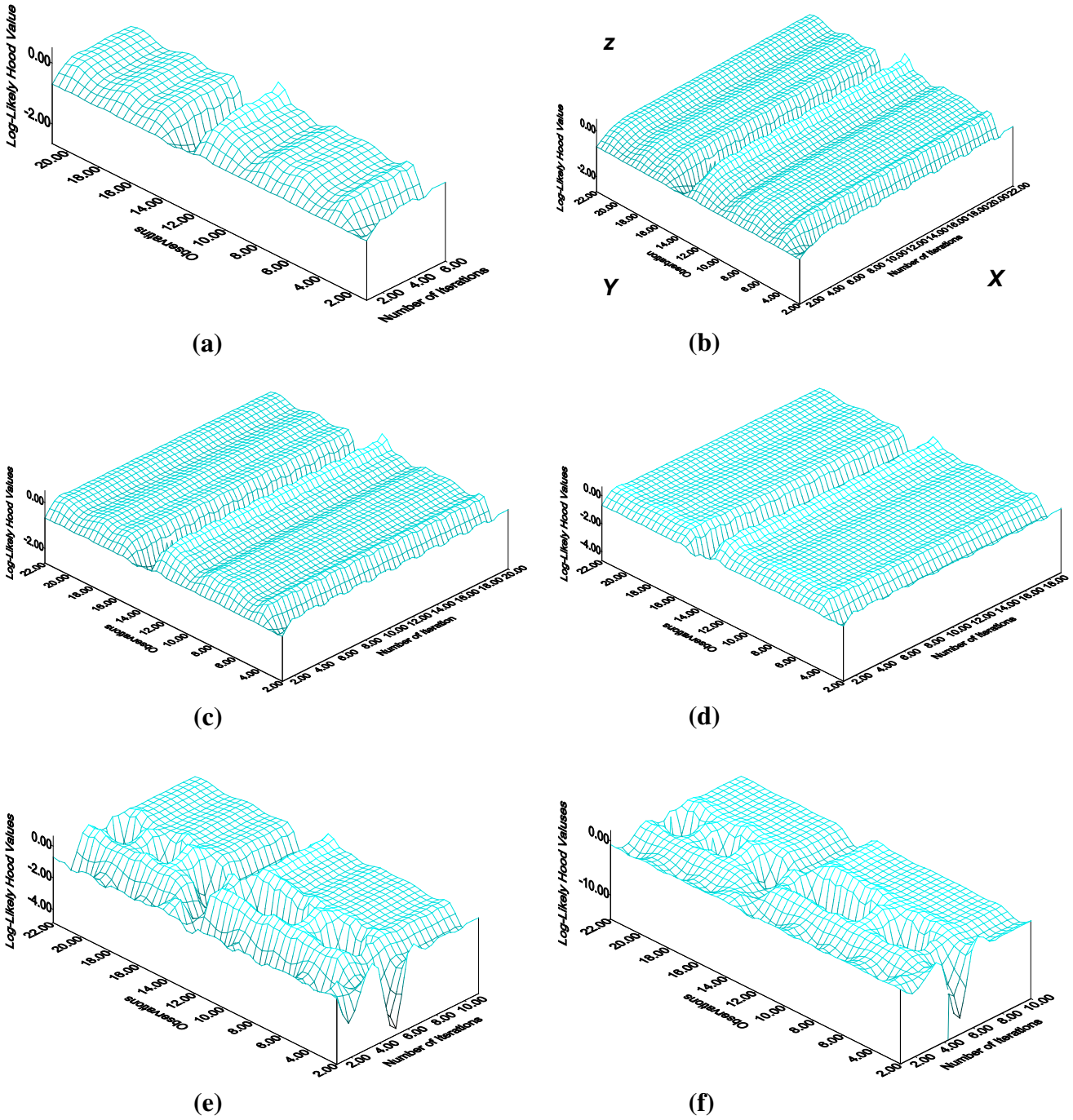

Fig. 5 Estimation surface converged for six algorithms with $CR = 10^{-4}$, **a** Newton–Raphson ($\lambda = 1$), **b** BHHH ($\lambda = 1/2$), **c** BHHH-2 ($\lambda = 1/2$), **d** Steepest Ascent ($\lambda = 16$), **e** DFP ($\lambda = 16$), **f** BFGS ($\lambda = 8$)

convergence behaviour of estimation algorithms determined by Hessian updating mechanism prefixed in each algorithm can be clearly demonstrated. Log-likelihood values are presented on *Z*-axis; the number of iterations is on *X*-axis, and observation numbers for estimation sample are on *Y*-axis (Fig. 5b). In Fig. 5a, log-likelihood value is calculated for each observation from the first iteration and it showed the same value (−0.693147181). This value resulted from the assumption that all parameter values are zero in the initial stage of estimation in the model specification. NR, as shown in Fig. 5a, shows the most simple convergence behaviour in terms of small number of iteration (*X*-axis), meaning that short convergence time. The log-likelihood value for NR was stared with −14.55609079 and converged with the value of −6.166042212 at 6th run. Algorithms performance for DFP and BFGS are shown in Fig. 5e, f respectively. It can be easily seen, that their convergence pattern are similar in terms of both the number of iterations and convergence time. BHHH and BHHH-2 are presented in Fig. 5b, c respectively and are also very similar in their convergence behaviour. These similarities are the result of using almost same iterative process in terms of updating approximate Hessian matrix. SN shown in Fig. 5d, showed similar performance with BHHH and BHHH-2.

5.1 Numerical experiment 1

The two different levels of convergence criteria used in this experiment are used from two earlier studies [5, 8, 9]. The iteration is completed with $\beta_{t+1,k}$ as the solution, when the prefixed stopping precision is satisfied. The mathematical expressions used to represent convergence criteria are shown below. Numerical experiment1 generally uses $\beta_0 = [\beta_{01} = 0, \beta_{02} = 0]'$, as initial guessing of starting points.

$$\left[\frac{1}{k}\sum_{k=1}^{k}(\beta_{t+1,k} - \beta_{t,k})^2\right]^{1/2} < CR = 10^{-4}, \tag{38}$$

$$\left[\frac{1}{k}\sum_{k=1}^{k}(\beta_{t+1,k} - \beta_{t,k})^2\right]^{1/2} < CR = 10^{-6}. \tag{39}$$

5.1.1 Experimental run with $CR = 10^{-4}$

The result from the experimental runs (trials with convergence) based on convergence criteria ($CR = 10^{-4}$) for six different algorithms are summarized graphically in Figs. 6 and 7. From these figures, it is conclusive that the NR algorithm is the most appropriate in terms of all three performance measures despite its mathematical difficulty to compute the real Hessian of the log-likelihood function. The performance results based on for one iteration time are found less than 1 s because of a simple model specification. We tried with varied step sizes for all six algorithms to obtain the maximum value for the log-likelihood function. The step size can be considered at random during the initial stage of estimation. NR with step size $\lambda = 1$ showed better performance (converged after 6 iterations) than the other algorithms as far as the number of iteration criteria is concerned.

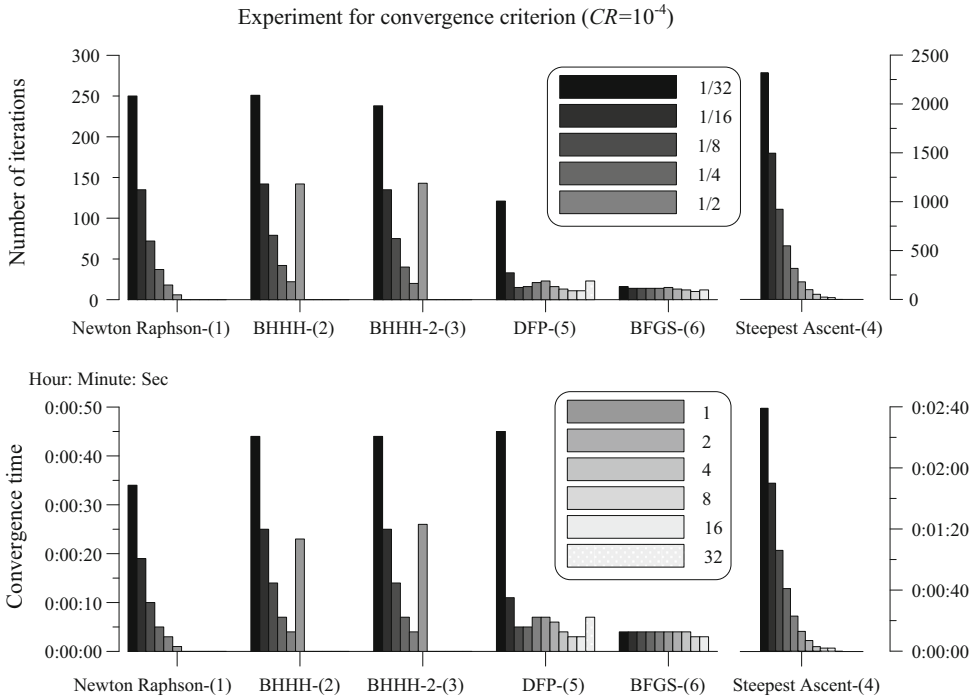

Fig. 6 Experimental run results of six algorithms with ($CR = 10^{-4}$)

The convergence time can be used as a sign of the operational efficiency of an algorithm because it shows critically the relative superiority in convergence speed of an algorithm. In the case of number of iterations, there are step size differences in the same algorithm; also differences between the six algorithms (Fig. 6). The SA algorithm showed the worst performance. The application of the convergence time showed notable differences between some algorithms. The NR algorithm showed the best performance with respect to the convergence time for all step sizes, followed by the BFGS, DFP and BHHH, BHHH-2, and SA algorithm in order. Particularly, with step size λ

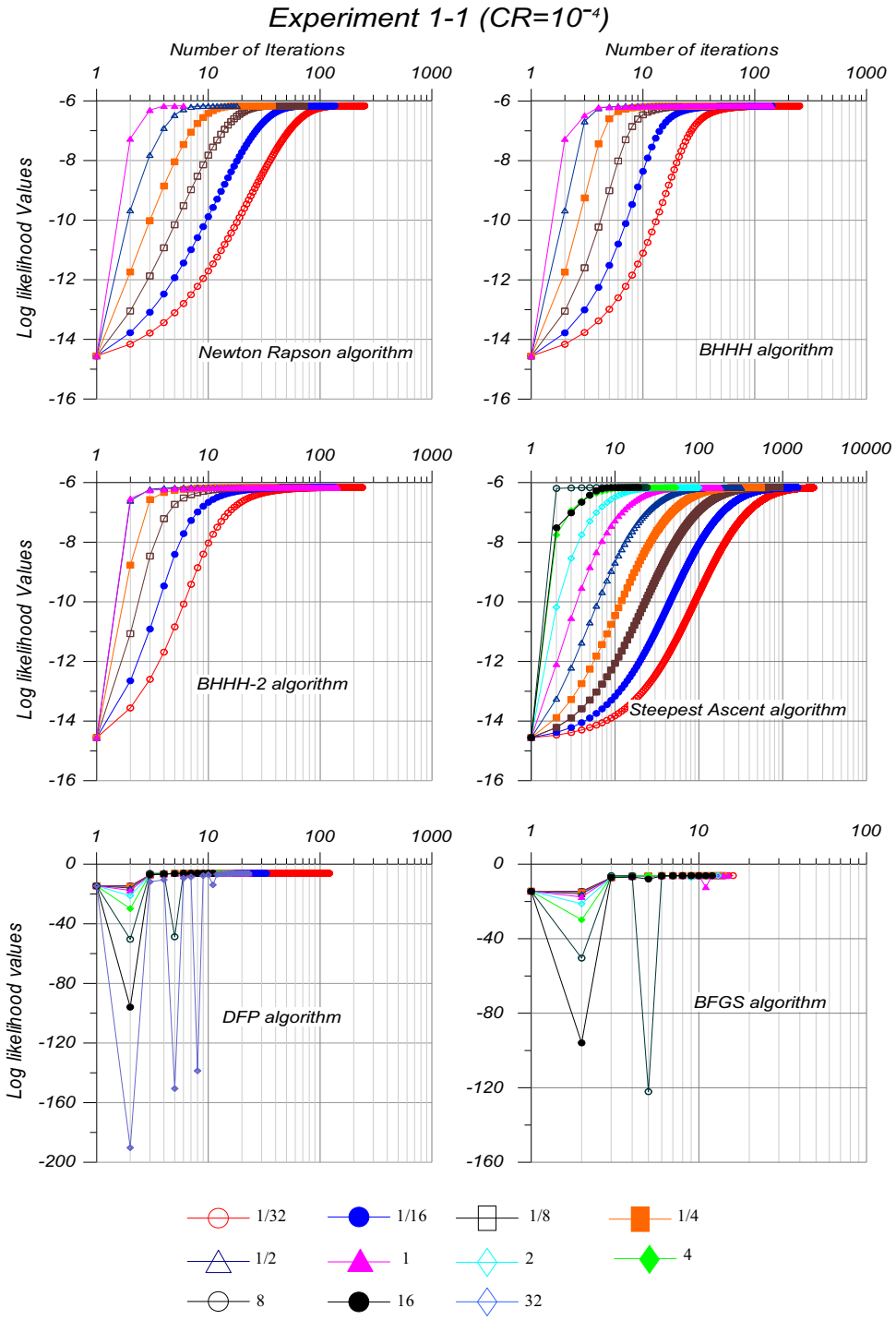

Fig. 7 Illustration of variation of Log-likelihood values in iteration process

equal to 1, the NR algorithm needs just 1 s to finish calculation. Conversely, SA algorithm takes 2 min 39 s with the step size $\lambda = 1/32$.

5.1.2 Experimental run with $CR = 10^{-6}$

In experiment for the evaluation of performance with different convergence criteria ($CR = 10^{-6}$), a different precision level of stopping convergence criteria having the same mathematical expression as shown in Eq. 39 is used. The comparison results are presented graphically in Fig. 8. For all algorithms, the estimation performance has deteriorated as compared to experiment conducted with $CR = 10^{-4}$ in terms of all two measures. These deteriorations took place for all algorithms, step sizes, and performance measures. For example, in the case of SA algorithm ($\lambda = 1/32$), the experiment convergence time $CR = 10^{-6}$ increased significantly (from 0:02:39 to 0:09:43), also the difference of the number of iterations is significant (from 2,320 to 7,033). The pattern of deteriorations is generally the same for all six algorithms. However, NR algorithm required only seven iterations for step size 1. In the run using $CR = 10^{-6}$ with a step size one, one iteration number is increased as compared to the run using $CR = 10^{-4}$ with a step size one. The convergence behaviour is identical to experiment using $CR = 10^{-4}$ except the fact that slight differences occur for all two performance measures in terms of numerical values.

5.2 Numerical experiment 2

The second numerical experiment is intended for the examination of variations in algorithm performance, which may arise by considering various starting points for the parameters. This experiment is initiated to understand how to guess the starting points, and analyze the prospective of the different estimation performances. Nonetheless, searching a new suitable method for a good initial guess of the starting points is not within the scope of this experiment. In this study, the focus is to know that, whether it can make a difference in estimating performance by changing initial starting points, particularly in custom built environment. The outcome of this experiment will help in suggesting a new methodical approach of guessing starting points for a better operational performance. This may be considered as future research of this study. This experiment is designed to in line of this new interesting topic. In this investigation, another vector of parameter is introduced: $\beta_0 = [\beta_{01} = -0.1, \beta_{02} = -0.1]'$, in which two parameters are changed from 0 to −0.1 so as to reflect the case, where an analyst has prior knowledge of the parameter estimates and a negative (−) sign can be assigned to the two parameters. The convergence criterion remains same as it is given in subsect. 5.1.1. Detailed results for all algorithms are presented in Fig. 9. Overall, a little improvements were observed in terms of all performance but, in a few cases, deterioration has also happened. Taking

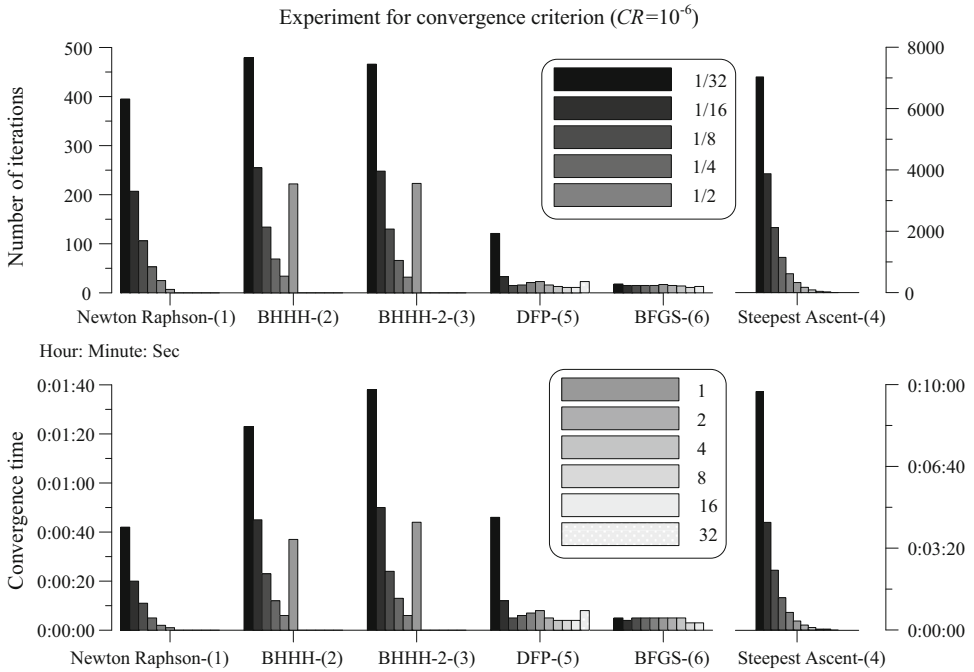

Fig. 8 Experimental run results of six algorithms with $CR = 10^{-6}$

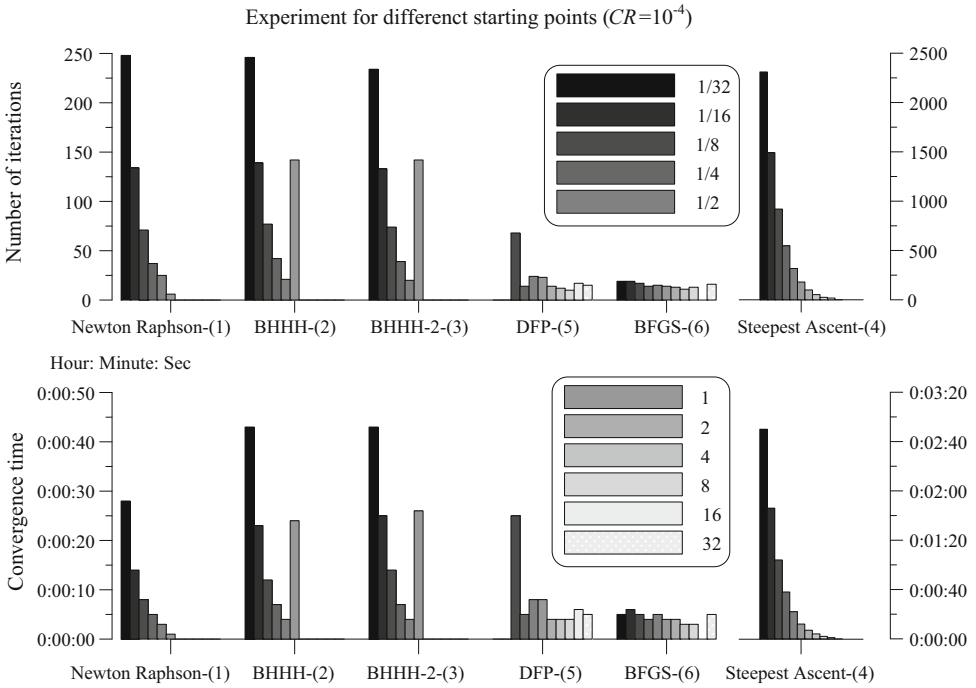

Fig. 9 Experimental run results of six algorithms with different starting points

the example of DFP algorithm, a run with step sizes 1/32 and 1/16 did not converge with new starting points, however the same run converged in experiment with ($CR = 10^{-4}$ and 10^{-6}). The rationale for this dissimilar convergence behaviour is not definite in this study.

6 Conclusion and discussion

Discrete choice analysis has been of interest to researchers in many diverse disciplines for consumer's choice behaviour prediction. Particularly, in transportation domain, there have been many applications of discrete choice modeling techniques to predict the behaviour of users consuming transportation systems or services. However, almost all research associated with discrete choice modeling focuses only on the interpretation of the estimated results or on the efforts to find policy implications in model parameters to support decision making process. Furthermore, almost all types of discrete choice models are carried out using commercial statistical packages. In the earlier generation of discrete choice modeling, it was common practice to follow instructions of a selected specific computer package. However, as the choice model specification becomes more complicated and custom-built modeling is increasingly in demand, researchers working in the modeling domain are challenged to go beyond the first generation modeling practices. Nonetheless, there is a lack of literature available to research on estimation algorithms as well as important components in it. Although, some literatures on this may be available in econometric field [8–10], they are not directly related to the concerns of transportation modellers. Also, these studies have a different theoretical base, which puts some constraints to write the computer codes for transportation modeling application.

Several factors that dominate estimation performance are detected through this experimental model estimation. These are: (a) convergence criterion; (b) initial guessing of starting points. In this research a convergence criterion of $\left[\frac{1}{k}\sum_{k=1}^{k}\left(\beta_{t+1,k}-\beta_{t,k}\right)^2\right]^{1/2}$, is used, which is generally accepted convergence criterion in many literatures [8, 9]. Based on the results, some important aspects might be considered in more elaborated research associated with custom-built modeling. Possibly, potential research extensions are noted below.

(a) Different mathematical formulations for stopping criterion may be tried and different level of stopping accuracy such as $\left[\frac{1}{k}\sum_{k=1}^{k}\left(\beta_{t+1,k}-\beta_{t,k}\right)^2\right]^{1/2} < 10^{-6}$, 10^{-7} can be adopted to make a significant difference in model estimation performance. This issue should be revisited in the future work.

(b) When researchers have to make a guess on initial parameters starting points in custom-built modeling, it would be time saving to make an initial guess of model's parameters fairly close to true values on the first trial. However almost all choices of initial guess of parameters

set up by researchers are not the actual parameter estimates of the given model. In this context, it is a challenge for researchers to be more organized in choosing a vector of parameters, which may be a critical subject in custom-built modeling. This suggests that initial guessing of parameters can affect a model estimation performance to a high extent in terms of iteration numbers, convergence time and so on. A similar research study was reported by Liu and Mahmassani [11] that uses an empirical model estimation process for obtaining parameters of a probit model. They use the genetic algorithm for making a guess of initial parameter estimates. The authors of this paper have demonstrated that initial guessing of parameter can have significant effect on overall model estimation performance. All these issues addressed here are note worthy for future extension of research reported in this paper.

References

1. Hyuk-Jae R (2013) Mode choice behavior of various airport user groups for ground airport access. Open Transp J 7:43–55
2. Hyuk-Jae R (2013b) Empirical application of four fundamental algorithms for parameter estimates of multinomial logit discrete choice model. In: CSCE conference, Montreal
3. Kenneth T (2003) Discrete choice methods with simulation. Acadmic Press, Cambridge
4. Hyuk-Jae R, Ata MK (2013) Enhancing algorithmic base for discrete choice modelling. KSCE J Civ Eng 17(7):1798–1809
5. Moshe BA, Steven RL (1985) Discrete choice analysis: theory and application to travel demand. MIT Press, Cambridge
6. Green William H G (2003) Econometric analysis. Pearson Education, Inc, Upper saddle river
7. Daniel LM (1974) Conditional logit analysis of qualitative choice behaviour. Frontiers in econometrics. Academic press, New York
8. David AB (1979) On the computational competitiveness of full-information maximum likelihood and three-stage least-squares in the estimation of nonlinear, simultaneous-equations models. J Econom 9:315–342
9. David AB (1980) On the efficient computation of the nonlinear full-information maximum-likelihood estimator. J Econom 14:203–225
10. David SB (1988) A comparison of algorithms for maximum likelihood estimation of choice models. J Econom 38:145–167
11. Yu-Hsin L, Hani SM (2000) Global maximum likelihood estimation procedure for multinomial probit (MNP) model parameters. Transp Res Part B34:419–449